(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
9 August 2001 (09.08.2001)

PCT

(10) International Publication Number
WO 01/57651 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/US01/03740

(22) International Filing Date: 5 February 2001 (05.02.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/499,273 7 February 2000 (07.02.2000) US

(71) Applicant *(for all designated States except US)*: **ATTACHMATE CORPORATION** [US/US]; 3617-131st Avenue SE, Bellevue, WA 98006 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **PROVO, Nathan, J.** [US/US]; 25533 223rd Court SE, Maple Valley, WA 98038 (US).

(74) Agents: **JOHNSON, Brian, L.** et al.; Seed Intellectual Property Law Group PLLC, Suite 6300, 701 Fifth Avenue, Seattle, WA 98104-7092 (US).

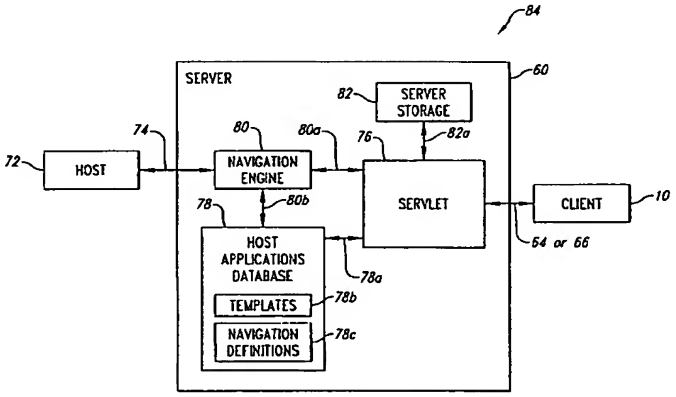
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR USER INTERFACE TRANSLATION



(57) Abstract: A system and method for user interface translation utilizes a server computer that communicates with a legacy host computer and a browser operating on a client computer. The server computer includes a servlet, host applications database, navigation engine, and server storage. Before operation the host applications database is populated with data for transformation templates and navigation definitions. The transformation templates include details describing how non-markup language based data structures will be translated for display under the markup language based browser by a monitor of the client computer. The navigation definitions contain keystroke data used as input to a non-markup language based user terminal for navigation between display screens generated and transmitted to the non-markup language based user terminals from the legacy host computer. The server computer receives commands with uniform resource locator arguments from the client computer and retrieves an appropriate navigation definition for the navigation engine to establish a terminal session with the legacy host computer and a requested host application. The server computer receives non-markup language based data structure responses from the host application and translates these responses into markup language data structures for display on the client computer.

WO 01/57651 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR USER INTERFACE TRANSLATION

TECHNICAL FIELD

The invention relates generally to user interfaces to access host applications, and more particularly to a system and method to use mark-up language based graphical user interfaces to access terminal based host applications.

BACKGROUND OF THE INVENTION

Markup computer languages, such as HyperText Markup Language (HTML), have in many cases greatly improved the ease of use of user interfaces of client computers utilized to access data and applications via computer networks such as the Internet. Markup languages use syntactically delimited characters added to the data of a document to represent the structure of the document. Approaches for markup include descriptive markup tags, references, markup declarations, and processing instructions. By using markup language based user interface browser programs to access and display markup language based data and applications found on markup language based server computer systems, the markup language based data and applications can be presented in more effective ways than available with non-markup language based user terminals accessing non-markup language based legacy host computer systems.

Much of data and applications, however, remain on non-markup language based legacy host systems that are accessed by non-markup language based user terminals. Conventional markup language based browsers have allowed for running non-markup language based terminal emulation programs within display windows of the markup language based browsers. Unfortunately, these non-markup language based terminal emulation programs display non-markup language based data and applications from the non-markup language based legacy host systems in a form, which does not provide the improvements associated with the markup language based browsers when accessing data and applications on markup language based server systems. Moving the data and applications found on non-markup language based legacy

host systems to markup language based server systems to be accessed by markup language based browsers has not been economically feasible in many cases. As users become more accustomed to the ease of use provided by the markup language based browsers, reliance upon non-markup language based user terminals and terminal emulation programs to access data and applications on non-markup language based legacy host systems becomes increasingly frustrating.

SUMMARY OF THE INVENTION

The present invention resides in a system and method for user interface translation. Aspects of the system and method include a server system for use with a legacy host computer and a client computer. The legacy host computer is configured to run host applications that produce non-markup language based data structures and communicate with non-markup language based user terminals that display only non-markup language based data structures. The client computer includes markup language based software and a monitor. The markup language based software is configured to direct display of markup language data structures on the monitor. The client computer is configured to transmit commands associated with uniform resource locator (URL) designations.

The server system includes a host applications database configured to store one or more transformation templates associated with non-markup language based data structures received by a navigation engine from host applications running on the legacy host computer. The transformation templates are configured to provide details used in generating markup language based data structures for display on the client computer. The host applications database is configured to store navigation definitions containing keystroke data used as input to non-markup based user terminals and transmitted to the host applications to navigate between screens displayed by the non-markup based user terminals generated from non-markup language based data structures sent from the host applications to the non-markup language based user terminals.

The server system further includes the navigation engine communicatively linked to the legacy host computer via a first communication link.

The first communication link includes a first terminal session to a first host application of the host applications running on the legacy host computer. The navigation engine is configured to request and receive a non-markup language based data structure produced by the first host application from the legacy host computer via the first terminal session of the first communication link based on one or more of the navigation definitions stored on the host applications database.

The server system also includes a servlet configured to send markup language based data structures to the client computer based upon the non-markup language based data structures received by the navigation engine. The servlet is communicatively linked to the client computer to receive the commands associated with URL designations. The servlet is configured to have the navigation definitions sent to the navigation engine based upon the commands associated with URL designations received from the client computer.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a computing system suitable for employing aspects of the invention to store, transmit and display portions of web pages.

Figures 2 is a block diagram illustrating detail of the server computer shown in Figure 1.

Figure 3 is an exemplary block diagram illustrating navigation between four screens as subject matter of a navigation definition.

Figure 4 is a communication diagram illustrating start-up communication between a client and a server of an embodiment of the present invention.

Figure 5 is a communication diagram illustrating communication involved with an initial request to display an application for an embodiment of the present invention.

Figure 6 is a communication diagram illustrating communication involved with a request for display of an additional screen of a displayed application for an embodiment of the present invention.

Figure 7 is a communication diagram illustrating communication involved with a request for update of an additional screen of a displayed application for an embodiment of the present invention.

Figure 8 is a communication diagram illustrating communication involved with a request to close an open application for an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

A system and related method for user interface translation is described. In the following description, numerous specific details are provided, such as specific types of displayable information structures, network architectures, storage methods, etc., to provide a thorough understanding of embodiments of the invention. One skilled in the relevant art, however, will recognize that the invention can be practiced without one or more of these specific details, or with other displayable information structures, network architectures, storage methods, etc. In other instances, well-known structures or operations are not shown, or not described in detail, to avoid obscuring aspects of the invention or for brevity.

Figure 1 and the following discussion provide a brief, general description of a suitable computing environment in which the invention can be implemented. Although not required, embodiments of the invention will be described in the general context of computer-executable instructions, such as program application modules, objects, or macros being executed by a personal computer. Those skilled in the relevant art will appreciate that the invention can be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, mini computers, mainframe computers, and the like. The invention can be practiced in distributed computing environments where tasks or modules are performed by remote processing devices, which are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Referring to Figure 1, a conventional personal computer referred herein as a client computer 10 includes a processing unit 12, a system memory 14 and a system bus 16 that couples various system components including the system memory to the processing unit. The processing unit 12 may be any logic processing unit, such as one or more central processing units (CPUs), digital signal processors (DSPs), application-specific integrated circuits (ASIC), etc. Unless described otherwise, the construction and operation of the various blocks shown in Figure 1 are of conventional design. As a result, such blocks need not be described in further detail herein, as they will be understood by those skilled in the relevant art.

The system bus 16 can employ any known bus structures or architectures, including a memory bus with memory controller, a peripheral bus, and a local bus. The system memory 14 includes read-only memory ("ROM") 18 and random access memory ("RAM") 20. A basic input/output system ("BIOS") 22, which can form part of the ROM 18, contains basic routines that help transfer information between elements within the client computer 10, such as during start-up.

The client computer 10 also includes a hard disk drive 24 for reading from and writing to a hard disk 25, and an optical disk drive 26 and a magnetic disk drive 28 for reading from and writing to removable optical disks 30 and magnetic disks 32, respectively. The optical disk 30 can be a CD-ROM, while the magnetic disk 32 can be a magnetic floppy disk or diskette. The hard disk drive 24, optical disk drive 26 and magnetic disk drive 28 communicate with the processing unit 12 via the bus 16. The hard disk drive 24, optical disk drive 26 and magnetic disk drive 28 may include interfaces or controllers (not shown) coupled between such drives and the bus 16, as is known by those skilled in the art. The drives 24, 26 and 28, and their associated computer-readable media, provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for the client computer 10. Although the depicted client computer 10 employs a hard disk, optical disk 30 and magnetic disk 32, those skilled in the relevant art will appreciate that other types of computer-readable media that can store data accessible by a computer may be

employed, such as magnetic cassettes, flash memory cards, digital video disks ("DVD"), Bernoulli cartridges, RAMs, ROMs, smart cards, etc.

Program modules can be stored in the system memory 14, such as an operating system 34, one or more application programs 36, other programs or modules 38 and program data 40. The system memory 14 also includes a browser 41 for permitting the client computer 10 to access and exchange data with web sites of the Internet, corporate intranets, or other networks as described below, as well as legacy host computers as found in the depicted embodiment of the invention. The browser 41 is markup language based and operates with markup languages that use syntactically delimited characters added to the data of a document to represent the structure of the document. As described in detail below, markup language based documents and other markup language based data structures are translated from non-markup language based data structures for displayed by the browser 41. The non-markup language based data structures are generated by computer applications originally designed to be displayed as host screens on non-markup language based user terminals. In the depicted embodiment, the browser operates with HyperText Markup Language (HTML) and also uses extensible stylesheet language (XSL) to convert extensible markup language (XML) documents into HTML documents.

While shown in Figure 1 as being stored in the system memory 14, the operating system 34, application programs 36, other modules 38, program data 40 and browser 41 can be stored on the hard disk of the hard disk drive 24, the optical disk 30 of the optical disk drive 26 and/or the magnetic disk 32 of the magnetic disk drive 28. A user can enter commands and information into the client computer 10 through input devices such as a keyboard 42 and a pointing device such as a mouse 44. Other input devices can include a microphone, joystick, game pad, scanner, etc. These and other input devices are connected to the processing unit 12 through an interface 46 such as a serial port interface that couples to the bus 16, although other interfaces such as a parallel port, a game port or a universal serial bus ("USB") can be used. A monitor 48 or other display device is coupled to the bus 16 via a video interface 50, such as a video

adapter. The client computer 10 can include other output devices, such as speakers, printers, etc.

The client computer 10 can operate in a networked environment using logical connections to one or more remote computers, such as a server computer 60. The server computer 60 can be another personal computer, a server, or other type of computer, and typically includes many or all of the elements described above for the client computer 10. The server computer 60 is logically connected to the client computer 10 under any known method of permitting computers to communicate, such as through a local area network ("LAN") 64 or a wide area network ("WAN") or the Internet 66. Such networking environments are well known in enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the client computer 10 is connected to the LAN 64 through an adapter or network interface 68 (coupled to the bus 16). When used in a WAN networking environment, the client computer 10 often includes a modem 70 or other device for establishing communications over the WAN/Internet 66. The modem 70 is shown in Figure 1 as coupled between the interface 46 and the WAN/Internet 66. In a networked environment, program modules, application programs, or data, or portions thereof, can be stored in the server computer 60. In the depicted embodiment, the client computer 10 is linked to the server computer through the LAN 64 or WAN/Internet 66 with TCP/IP middle layer network protocols and HTTP upper layer network protocols; however, other similar network protocol layers are used in other embodiments. The server computer 60 is further linked through a communication link 74 to a non-markup language based legacy host computer 72 such as an International Business Machines (IBM) host computer or other host computer such as Digital Equipment Company (DEC) host computer, Hewlett Packard host computer, Wang host computer, Sperry Univac host computer, Data General host computer or other such host computers running non-markup language based host applications 72a with non-markup language based data 72b. The legacy host computer 72 is conventionally accessed by non-markup language based user terminals; however, in the depicted embodiment of the invention, access is provided to the markup language

based browser 41 operating on the client computer 10 as explained in further detail below.

In the depicted embodiment, the communication link 74 can be any direct hardwire connection through a network device such as a terminal server or a virtual network connection such as a persistent TCP/IP socket connection that supports terminal sessions such as a telnet session of the TCP/IP suite. The upper communication layers of the communication link 74 involve standards associated with terminals such as IBM 3270, 5250, or DEC VT 100 or 220, other ASCII based terminals or other similar terminal communication standards. Those skilled in the relevant art will readily recognize that the network connections shown in Figure 1 are only some examples of establishing communication links between computers, and other links may be used, including wireless links.

As shown in Figure 2, the depicted embodiment the server computer 60 further includes a servlet 76 communicatively linked to a host applications database 78 by communication link 78a, to a navigation engine 80 by communication link 80a, and to a server storage 82 by communication link 82a, which collectively form a user interface translation system 84. Generally, during operation of the server computer 60, the servlet 76, the host applications database 78, the navigation engine 80, and the server storage 82 are located in a portion of systems memory of the server computer. The servlet 76 acts as an interface for the server computer 60 to handle requests from and responses to the client computer 10 via Hypertext Transfer Protocol (HTTP) over the LAN 64 or the WAN/Internet 66.

In the depicted embodiment, the servlet 76 is constructed as a Java servlet, but other embodiments use other languages and constructs such as used by Active Server Pages. In the depicted embodiment, the servlet 76 receives requests containing uniform resource locator (URL) arguments from the browser 41 running on the client computer 10. The servlet 76 generates responses to requests by the browser 41 by interacting with other linked resources of the user interface translation system 84 including the host applications database 78, the navigation engine 80, the server storage 82, and the legacy host computer 72.

Before the user interface translation system 84 can be put into operation, the host applications database 78 containing transformation templates 78b and navigation definitions 78c must be populated with details concerning the host applications found on the legacy host computer 72. Regarding the navigation definitions 78c contained in the host applications database 78a, recording tools found in the art are first used to observe and record actual keystroke data required for input into keyboards or other such devices of non-markup language based user terminals (not shown) to navigate between screens provided by the non-markup language based host applications 72a operating on the legacy host computer 72.

For instance, the example provided in Figure 3 has four host application screens for an exemplary non-markup language based host application 72a with navigation paths between a MainMenu screen 92, and three other screens: Logon screen 90, Calendar screen 94, and Mail screen 96. As part of the recording process for this example, a navigation path 98 from the Logon screen 90 and the MainMenu screen 92 would be observed to carry the keystroke input data "user1[tab]pswd[E]," which would be recorded and associated with the navigation path 98 as part of the navigation definition 78c for this non-markup language based host application 72a by a conventional recording tool.

Further navigation details of the navigation definitions 78c for the host applications database 78 include assigning a name to each host application screen (such as "Logon" for the Logon screen 90), assigning names to each field of a host screen (such as "userid" for userid field 90b of the Logon screen 90), and designating which static fields are to identify a screen (such as "Vtam" for the Logon screen 90). All these navigation details are included into the navigation definition for a particular host application 72a and stored in the host applications database 78 on the server computer 60.

A second preliminary action before the user interface translation system 84 can be put into operation involves determining how the transformation template 78b will transform each host screen generated by the non-markup language based host applications 72a with the host screens being originally designed for display on non-

markup language based user terminals but now being transformed for display through the markup language based browser 41. The transformation templates 78b are used to define how the transformed host screens as markup language based documents will appear on the monitor 48 of the client computer 10 as directed by the browser 41. In the depicted embodiment, the browser 41 displays HyperText Markup Language (HTML) documents so that the transformation template 78b of each host screen can include the full scope of display capability under the browser 41 with the HTML language. For instance, text from fields on a screen of a host application can be sized as desired and can be assigned to any position location on the corresponding HTML document that will be displayed on the monitor 48 of the client computer 10 as directed by the browser 41. Furthermore, links, submit buttons, or other controls as described by the HTML language can be included in the transformation template 78b used for navigation to other screens of the same host application.

Once the navigation definition 78c and transformation template 78b are entered into the host applications database 78 as a host application package for a particular host application 72a, the user interface translation system 84 is able to support that particular host application. Any changes made to the fundamental layout of a host screen (as opposed to mere changes in data found in the fields of the host screen) should be reflected appropriately in the corresponding host application package containing the associated navigation definition 78c and transformation template 78b for the changed host screen found in the host applications database 78. Any changes that are desired regarding how a particular host screen looks on the browser 41 can be made to the transformation template 78b of the host application package corresponding to the particular host application 72a regardless of whether the particular host application has changed.

In the depicted embodiment, extensible stylesheet language (XSL) is used to define the transformation templates 78b. According to XSL usage, an extensible markup language (XML) document is used as input to an XSL transformation template 78b. Browsers, such as Microsoft Internet Explorer 5.0, can receive an XML document containing host application field data along with an XSL

transformation template 78b associated with the XML document and convert the XML document and XSL transformation template to an HTML document for display on the browser 41. Thus, in the depicted embodiment only XML data and associated XSL transformation templates 78b are transmitted from the server computer 60 to the client computer 10 without the need for transmission of larger HTML documents.

Since the combination of the XML document and the XSL transformation template 78b for a host screen is sent to the browser 41 for conversion into an HTML document, the server computer 60 does not have to perform conversions associated with the transformation template. Furthermore, the XSL transformation template 78b for a particular host screen need only be downloaded to the browser 41 of the client computer 10 once unless the fundamental layout of a host screen changes or if a change is desired for the look of a particular host screen as displayed on the browser 41. Since not all browsers support conversion of XML documents into HTML documents using XSL as described for the depicted embodiment, alternative embodiments of the server computer 60 send the browser 41 on the client computer 10 HTML documents or other markup language documents that the browser of the alternative embodiment can recognize. This alternative embodiment may require the server computer 60 to apply a transformation template 78b to generate an HTML document or other markup language document.

To illustrate some rudimentary operations of the depicted embodiment of the user interface translation system 84, Figures 4 - 8 contain diagrams of communication between components of the user interface translation system 84 involved in these operations. The servlet 76 receives commands from the browser 41 including get and post commands with URL arguments. These commands from the browser 41 are generally initiated by a user that activates a link, button, or other control displayed by the browser on the monitor 48. The get commands include requests for a main page that lists host applications 72a for further selection, requests for particular host applications, requests for navigation to particular screens of a host application, and requests for closing a particular application. The post commands include requests to

enter data into a presently viewed screen of one of the host applications 72a before navigating to another screen of the host application.

Initial entrance into the communication translation system 84 is done by sending get main page communication 120 from the browser 41 of the client 10 to the servlet 76 of the server computer 60 as shown in Figure 4. The main page includes a list of the host applications 72a on the legacy host computer 72 that are available for access by the browser 41 on the client computer 10. Part of the get main page communication 120 includes a URL associated with the main page.

Upon receipt of the get main page communication 120, the servlet 72 sends query database communication 122 to the host applications database 78, which involves iterating through the host applications database to generate host applications information in communication 124 sent from the host applications database 78 back to the servlet 76. Based upon the host applications information in communication 124, the servlet 76 builds a main page XML document that contains a name, description, and URL for each host application package found in the host applications database 78. The servlet 76 then requests a XSL transformation template 78b from the host applications database 78 in communication 126 to be used to transform the main page XML document into HTML for display by the browser 41 on the monitor 48.

In response, the host applications database 78 sends main XSL communication 128 containing the main page XSL transformation template 78b to the servlet 76. In generating the main page XML document, the servlet 76 adds an XSL tag to the XSL transformation template 78b. The main page XML document and the XSL transformation template 78b are then sent by the servlet 76 to the browser 41 on the client computer 10 in communication 130 where the browser uses the XSL transformation template to transform the main page XML document into an associated HTML document for display of the main page on the monitor 48.

After display of the HTML document associated with the main page, a user may select a particular host application for use from a list of URL links presented by the browser 41. As shown in Figure 5, once a selection of a host application "A" is made, a request application "A" communication 132 is sent by the browser 41 on the

client computer 10 to the servlet 76. The servlet 76, in turn, sends a request to the host applications database 78 for a navigation definition 78c for the host application "A" in communication 134 and receives the navigation definition 78c for the host application "A" in communication 136 as a reply. In the depicted embodiment, the navigation definition 78c is extracted by the servlet 76 from the appropriate host application package. Based upon the navigation definition 78c for the host application "A", the servlet 76 sends a request to the navigation engine 80 to establish a session for the host application "A" with the legacy host computer 72 and to navigate to a designated first screen of the host application "A" in communication 138. In an alternative embodiment, the servlet 76 commands the host applications database 78 to send the navigation definition 78c of session "A" directly to the navigation engine 80 via communication link 80b of Figure 2. Session "A" is established between the navigation engine 80 and the legacy host computer 72 in communication 140. The servlet 76 also sends reference to session "A" to the server storage 82 in communication 142 so that the session "A", once established, will remain persistant between the server computer 60 and the legacy host computer 72 until a command is sent from the browser 41 to close the connection. In the depicted embodiment, the session "A" reference is stored in a HttpSession object using the names of the host application 72a and the particular client computer 10 that initiated the session as identification for the session reference.

With the session "A" established between the legacy host computer 72 and the navigation engine 80 for the application "A", the servlet 76 sends a request through the navigation engine 80 to the legacy host computer 72 for field data of the first screen of the host application "A" in communication 144. The legacy host computer 72 replies by sending non-markup language based field data of the first screen of host application "A" through the navigation engine 80 to the servlet 76 in communication 146. The servlet 76 generates an XML document for the first screen of the host application " A" based on the non-markup language based field data of the first screen of the host application "A" received from the legacy host computer 72. In an alternative embodiment, the navigation engine 80 generates the XML document for the first screen of the host application "A" and sends the XML document to the servlet 76.

The servlet 76 then requests from the host applications database 78 an XSL transformation template 78b for the first screen of the host application "A" in communication 148 and receives the requested XSL transformation template in communication 150. The servlet 76 sends the XML document and XSL transformation template 78b for the first screen of the host application "A" to the browser 41 on the client computer 10 in communication 152. The browser 41 then uses the XSL transformation template 78b of communication 152 to convert the XML document of communication 152 into an HMTL document for the first screen of the host application "A" for display by the browser 41 on the monitor 48.

After viewing the HTML document for the first screen of the host application "A", a user may select an additional screen "A2" of the host application "A" by using the mouse 44 to activate a link displayed by the browser 41 on the monitor 48. In the depicted embodiment, activation of a link is implemented as a post if an update of the viewed screen is required before navigation to the next desired screen, otherwise activation of a link is implemented as a get. Activation of the link to host application "A2" as a get causes the browser 41 to send a request to get the screen "A2" of the host application "A" to servlet 76 in communication 160 of Figure 6.

Generally, links to additional screens of a particular host application will be displayed in screens of the same particular host application, so that the session for the particular host application will be open when another screen of the particular host application is requested. If the host application "A" has been previously selected for viewing before the screen "A2" is selected, the session "A" reference will remain in server storage 82 until the application "A" is closed. Details of the process to close an application are explained below. As provided by the example illustrated by Figure 5, the host application "A" was opened as part of communications 132 – 152. As part of the example illustrated by Figure 6, the host communication "A" continues to be in an open state so that the host application "A" is still open when communication 160 is sent. The servlet 76 will then retrieve the session "A" reference by sending communication 162 to the server storage 82. If the host application "A" is closed when the communication 160 is sent, then the host application "A" must be first opened

before proceeding to request the screen "A2" of the host application "A" for display on the browser 41.

Server storage 82 replies to communication 162 by sending the session "A" reference to the servlet 76 in communication 164. The servlet 76 then sends a request for screen "A2" of session "A" to the navigation engine 80 in communication 166. The navigation engine 80 then issues non-markup language based user terminal based commands to the legacy host computer 72 to navigate to the screen "A2" in communication 168. The servlet 76 sends a request through the navigation engine 80 to the legacy host computer 72 for non-markup language based field data of screen "A2" of the host application "A" in communication 170. The legacy host computer 72 replies by sending the non-markup language based field data of screen "A2" for the host application "A2" through the navigation engine 80 to the servlet 72 in communication 172.

The servlet 72 requests an XSL transformation template for the screen "A2" of the host application "A" by sending communication 174 to the host applications database 78. The host application database 78 replies by sending the XSL transformation template for the screen "A2" of the host application "A" to the servlet 72 in communication 176. Based upon the non-markup language based field data for the screen "A2" of the host application "A" received from the legacy host computer 72 through the navigation engine 80 in communication 172, the servlet 72 generates an XML document for the "A2" screen. In communication 178, the servlet 72 sends the XML document and the XSL transformation template for the "A2" screen of the host application "A" to the browser 41 of the client computer 10 where the browser generates an associated HTML document to be displayed by the browser on the monitor 48.

An update to a currently viewed screen of a host application is accomplished with a post command from the browser 41 before navigation to another screen is performed. As an example, communication 200 of Figure 7 requests to update screen "A2" of the host application "A" followed by navigation to the screen "A3" of the host application "A". Communication 200 sent from the browser 41 on the client computer 10 is implemented as a put command, in the depicted embodiment, which

includes field values for updating screen "A2" and getting screen "A3" as illustrated in Figure 7. In response to communication 200, the servlet 76 retrieves the session "A" reference from the server storage 82 in communications 202 and 204. The servlet 76 then requests the navigation engine 80 to update screen "A2" with the field values provided by the browser 41 and requests navigation to screen "A3" in communication 206. The navigation engine 80 updates the screen "A2" and navigates to the screen "A3" in communications 208 and 210, respectively, to the legacy host computer 72. The servlet 76 then requests and receives non-markup language based field data of screen "A3" of the host application "A" from the legacy host computer 72 through the navigation engine 80 in communications 212 and 214, respectively. The servlet 76 generates an XML document for the screen "A3" of the host application "A" from the non-markup language based field data received from the legacy host computer 72. The servlet 76 requests and receives an XSL transformation template 78b for the screen "A3" from the host applications database 78 in communications 216 and 218, respectively. The servlet 76 then sends the XML document and the XSL transformation template 78b to the browser 41 on the client computer 10 in communication 220. From the received XML document and XSL transformation template, the browser 41 generates an associated HTML document for display by the browser on the monitor 48.

To close the open host application "A", the browser 41 on the client computer 10 sends a request in communication 230 to servlet 76, as shown in Figure 8. Servlet 76 responds by requesting and receiving the session "A" reference from the server storage 82 in communications 232 and 234, respectively. The servlet 76 then commands the server storage 82 to remove the session "A" reference from storage in communication 236. The servlet 76 commands the navigation engine 80 to disconnect session "A" in communication 238 followed by the navigation engine commanding the legacy host computer 72 to disconnect session "A" in communication 240. The servlet 76 then sends a command to the browser 41 to redirect the display on the monitor back to main page in communication 242. If the browser 41 has retained both the XML document and the XSL transformation template 78b for the main page, the servlet 76 does not have to send them to the browser in communication 242.

From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.

CLAIMS

It is claimed:

1. A server system for use with a legacy host computer configured to run host applications that produce non-markup language based data structures and communicate with non-markup language based user terminals that display only non-markup language based data structures, the server system for use with a client computer including markup language based software and a monitor, the markup language based software configured to direct display of markup language data structures on the monitor, the client computer configured to transmit commands associated with uniform resource locator (URL) designations, the server system comprising:

a host applications database configured to store one or more transformation templates associated with non-markup language based data structures received by a navigation engine from host applications running on the legacy host computer, the transformation templates configured to provide details used in generating markup language based data structures for display on the client computer, the host applications database configured to store navigation definitions containing keystroke data used as input to non-markup based user terminals and transmitted to the host applications to navigate between screens displayed by the non-markup based user terminals generated from non-markup language based data structures sent from the host applications to the non-markup language based user terminals;

the navigation engine communicatively linked to the legacy host computer via a first communication link, the first communication link including a first terminal session to a first host application of the host applications running on the legacy host computer, the navigation engine configured to request and receive a non-markup language based data structure produced by the first host application from the legacy host computer via the first terminal session of the first communication link based on one or more of the navigation definitions stored on the host applications database; and

a servlet configured to send markup language based data structures to the client computer based upon the non-markup language based data structures received by the navigation engine, the servlet communicatively linked to the client computer to receive the commands associated with URL designations, the servlet configured to have the navigation definitions sent to the navigation engine based upon the commands associated with URL designations received from the client computer.

2. The communication system of claim 1 wherein the servlet converts the non-markup language based data structures into first markup language based data structures and the servlet sends the first markup language based data structures with the transformation templates to the client computer for conversion using the transformation templates by the markup language based software of the client computer into the markup language based data structures for display on the client computer.

3. The communication system of claim 1 wherein the servlet converts the non-markup language based data structures into the markup language based data structures for display on the client computer using the transformation templates.

4. The communication system of claim 1 wherein the navigation engine converts the non-markup language based data structures into first markup language based data structures, and the servlet sends the first markup language based data structures with the transformation templates to the client computer for conversion by the markup language based software of the client computer of the first markup language based data structures into the markup language based data structures for display on the client computer.

5. The communication system of claim 1 wherein the navigation engine converts the non-markup language based data structures into the markup

language based data structures for display on the client computer using the transformation templates.

6. The communication system of claim 1 wherein the servlet converts the non-markup language based data structures into extensible markup language (XML) based data structures, and the servlet sends the XML based data structures with the transformation templates for conversion by the markup language based software of the client computer of the XML based data structures into HyperText Markup Language (HTML) based data structures for display on the client computer.

7. The communication system of claim 1 wherein the transformation templates include details regarding positioning and sizing of text in the markup language based data structures generated for display on the client computer.

8. A server system for use with a legacy host computer configured to run host applications that produce non-markup language based data structures, the server system for use with a client computer including a browser program and a monitor, the browser program configured to receive extensible markup language (XML) data structures and to use extensible stylesheet language (XSL) transformation templates to convert the XML data structures into HyperText Markup Language (HTML) data structures, the browser program configured to direct display of the HTML data structures on the monitor, the server system comprising:

a navigation engine communicatively linked to the legacy host computer via a first communication link, the first communication link including a first persistent terminal session to a first host application of the host applications running on the legacy host computer, the navigation engine configured to receive a non-markup language based data structure produced by the first host application from the legacy host computer via the first persistent terminal session of the first communication link;

a host applications database server storage configured to store extensible stylesheet language (XSL) transformation templates associated with non-markup

language based data structures received by the navigation engine from host applications running on the legacy host computer; and

a servlet communicatively linked to the browser program of the client computer, the servlet configured to translate non-markup language based data structures received by the navigation engine into XML based data structures, the servlet configured to retrieve XSL transformation templates associated with the XML based data structures from the host applications database server and configured to transmit the XML based data structures and the XSL transformation templates to the browser program of the client computer via a second communication link.

9. A server system for use with a legacy host computer configured to run host applications that communicate in non-markup language based data structures with non-markup based user terminals that display only non-markup language based data structures, the server system for a client computer configured to transmit commands associated with uniform resource locator (URL) designations, the client computer configured to display markup language based data structures, the server system comprising:

a host applications database configured to store navigation definitions containing keystroke data used as input to the non-markup based user terminals and transmitted to the host applications to navigate between screens displayed by the non-markup based user terminals based upon non-markup language based data structures sent from the host applications to the non-markup based user terminals;

a navigation engine communicatively linked to the legacy host computer via a first communication link, the first communication link including a first persistent terminal session to a first host application of the host applications running on the legacy host computer, the navigation engine configured to receive a non-markup language based data structure produced by the first host application from the legacy host computer via the first persistent terminal session of the first communication link based on a navigation definition stored on the host applications database; and

a servlet communicatively linked to the client computer to receive the commands associated with URL designations, the servlet configured to have the navigation definitions sent to the navigation engine based upon the commands associated with URL designations received from the client computer.

10. A communication system for a legacy host computer running host applications, the legacy host computer configured to generate and transmit non-markup language based data structures to non-markup language based user terminals, the communication system comprising:

a client computer including a browser program and a monitor, the browser program configured to direct display of markup language based data structures on the monitor, the client computer configured to transmit client generated commands in a form recognizable by other than the legacy host computer; and

a server computer communicatively linked to the legacy host computer via a first communication link and communicatively linked to the client computer via a second communication link, the server computer configured to receive non-markup language based data structures from the legacy host computer via the first communication link, the server computer configured to translate the non-markup language based data structures into markup language based data structures, the server computer configured to transmit the markup language based data structures to the client computer via the second communication link, the server computer including a database of navigation definitions containing keystroke data used as input to the non-markup based user terminals and transmitted to host applications to navigate between screens displayed by the non-markup language based terminals generated from non-markup language based data structures sent from the host applications to the non-markup based user terminals, the server computer configured to receive the client generated commands from the client computer via the second communication link, the server computer configured to translate the client generated commands into keystroke data based upon the navigation definitions and to transmit the keystroke data to the legacy host computer via the first communication link.

11. A communication system for a legacy host computer configured to send non-markup language based data structures via a first communication link, the communication system comprising:

a client computer including a browser program and a monitor, the browser program configured to direct display of markup language based data structures on the monitor; and

a server computer communicatively linked to the legacy host computer via the first communication link and communicatively linked to the client computer via a second communication link, the server computer configured to receive non-markup language based data structures from the legacy host computer via the first communication link, the server computer configured to translate the non-markup language based data structures into markup language based data structures, the server computer configured to transmit the markup language based data structures to the client computer via the second communication link.

12. A communication system for a legacy host computer running host applications, the legacy host computer configured to generate and transmit non-markup language based data structures to non-markup language based user terminals, the communication system comprising:

a client computer configured to transmit client generated commands in a form recognizable by other than the legacy host computer; and

a server computer including a database of navigation definitions containing keystroke data used as input to non-markup based user terminals and transmitted to the host applications to navigate between screens displayed by the non-markup language based terminals generated from non-markup language based data structures sent from the host applications to the non-markup based user terminals, the server computer communicatively linked to the client computer via a first communication link and communicatively linked to the legacy host computer via a second communication link, the server computer configured to receive the client generated commands from the client computer via the first communication link, the

server computer configured to translate the client generated commands into keystroke data based upon the navigation definitions and to transmit the keystroke data to the legacy host computer via the second communication link.

13. A computer-readable medium for storing computer-readable instructions, the instructions written to program a server computer to perform a method, the method comprising:

receiving a first command from a client computer;

retrieving keystroke data based upon the first command, the keystroke data used as input to a non-markup based user terminal and transmitted to a non-markup language based host application to generate a host screen on the non-markup language based user terminal; and

sending the keystroke data to the host application.

14. A computer-readable medium for storing computer-readable instructions, the instructions written to program a server computer to perform a method, the method comprising:

receiving a non-markup language data structure from a legacy host computer; and

generating a markup language data structure based upon the non-markup language data structure and a transformation template, the transformation template containing details regarding how data of the non-markup language data structure is to be displayed by a markup language based browser as part of the markup language data structure.

15. A method of user interface translation, the method comprising:

recording keystroke data used as input to a non-markup based user terminal and transmitted to a non-markup language based host application to generate a host screen on the non-markup language based user terminal;

receiving a command from a client computer;

retrieving the recorded keystroke data based upon the command; and

sending the keystroke data to the host application.

16. The method of claim 15 wherein the command is associated with a uniform resource locator (URL).

17. A method of user interface translation, the method comprising:

receiving a non-markup language data structure from a legacy host computer;

retrieving a transformation template based upon the non-markup language data structure, the transformation template containing details regarding how data of the non-markup language data structure is to be displayed by a markup language based browser as part of a first markup language data structure; and

generating a second markup language data structure based upon the non-markup language data structure.

18. The method of claim 17, further comprising sending the second markup language data structure and transformation template to a client computer where the second markup language data structure is converted into the first markup language data structure based upon the transformation template.

19. The method of claim 17, further comprising generating the first markup language data structure from the second markup language data structure based upon the transformation template.

20. The method of claim 17, further comprising generating the second markup language data structure based upon the transformation template, the second markup language data structure being the first markup language data structure.

21. A method of user interface translation, the method comprising:

recording keystroke data used as input to a non-markup based user terminal and transmitted to a non-markup language based host application to generate a host screen on the non-markup language based user terminal;

receiving a command from a client computer;

retrieving the recorded keystroke data based upon the command,;

sending the keystroke data to the non-markup language based host application;

receiving a non-markup language data structure from a legacy host computer based upon the keystroke data sent to the host application;

retrieving a transformation template based upon the non-markup language data structure, the transformation template containing details regarding how data of the non-markup language data structure is to be displayed by a markup language based browser as part of a first markup language data structure; and

generating a second markup language data structure based upon the non-markup language data structure.

22. The method of claim 21, further comprising sending the second markup language data structure and transformation template to a client computer where the second markup language data structure is converted into the first markup language data structure based upon the transformation template.

23. The method of claim 21, further comprising generating the first markup language data structure from the second markup language data structure based upon the transformation template.

24. The method of claim 21, further comprising generating the second markup language data structure based upon the transformation template, the second markup language data structure being the first markup language data structure.

25. The method of claim 21 wherein the command is associated with a uniform resource locator (URL).

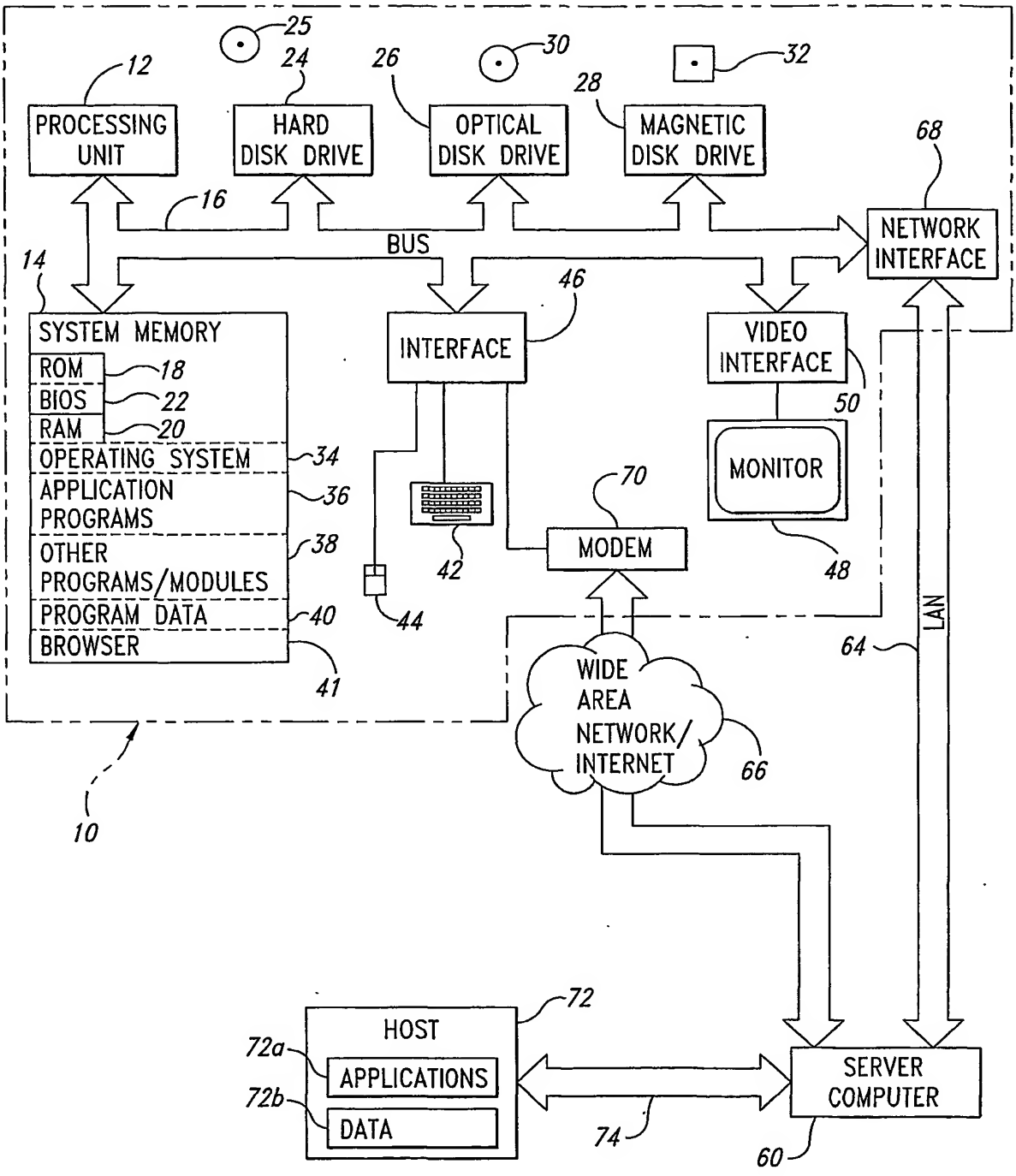

25
12
24
26
30
28
32
PROCESSING UNIT
HARD DISK DRIVE
OPTICAL DISK DRIVE
MAGNETIC DISK DRIVE
68
16
BUS
NETWORK INTERFACE
14
46
SYSTEM MEMORY
ROM
18
BIOS
22
RAM
20
OPERATING SYSTEM
34
APPLICATION PROGRAMS
36
OTHER PROGRAMS/MODULES
38
PROGRAM DATA
40
BROWSER
41
INTERFACE
VIDEO INTERFACE
50
MONITOR
48
70
MODEM
42
44
64
LAN
WIDE AREA NETWORK/ INTERNET
66
10
72
HOST
72a
APPLICATIONS
72b
DATA
74
SERVER COMPUTER
60

Fig. 1

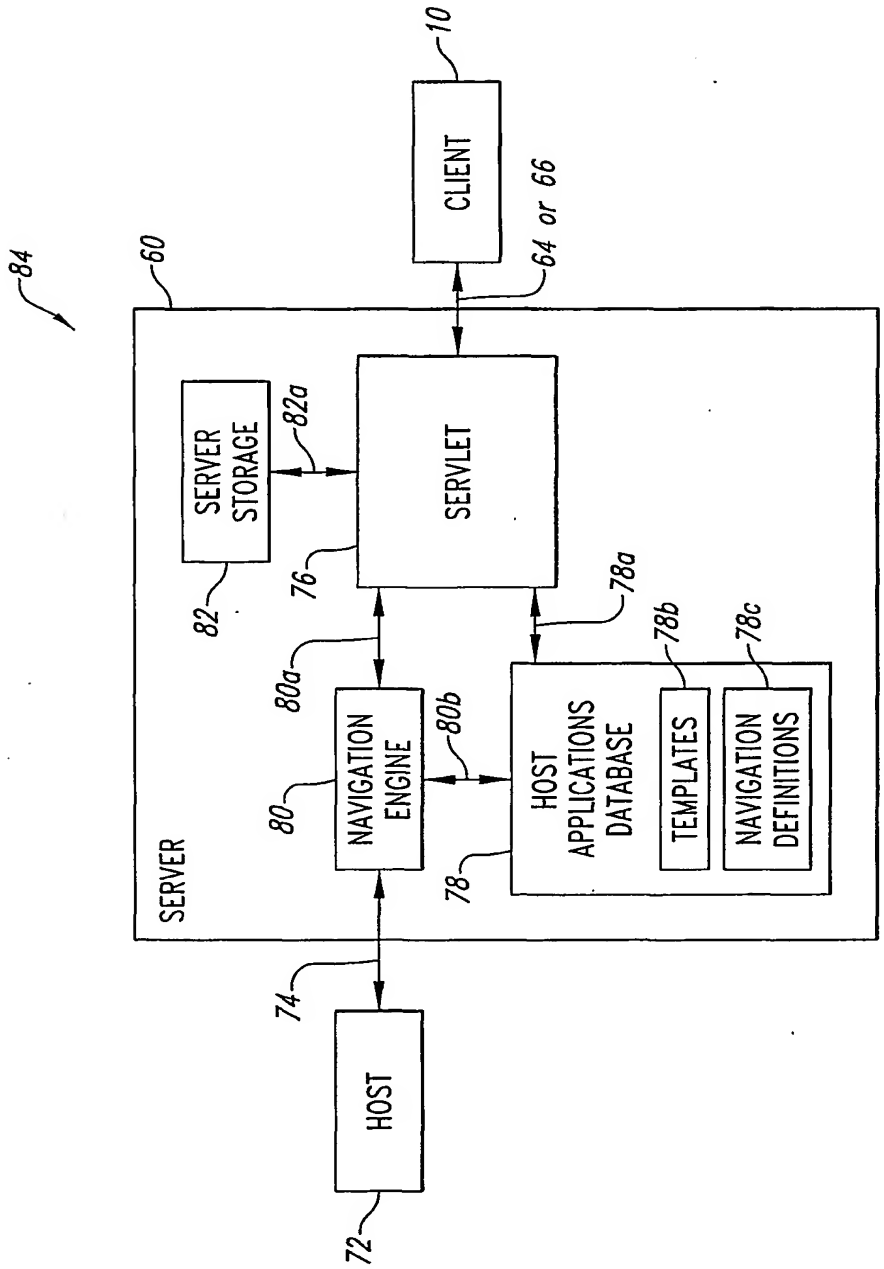

84
60
10
CLIENT
64 or 66
SERVER
SERVER STORAGE
82a
82
SERVLET
76
78a
78b
78c
80a
80
NAVIGATION ENGINE
80b
HOST APPLICATIONS DATABASE
TEMPLATES
NAVIGATION DEFINITIONS
78
74
HOST
72

Fig. 2

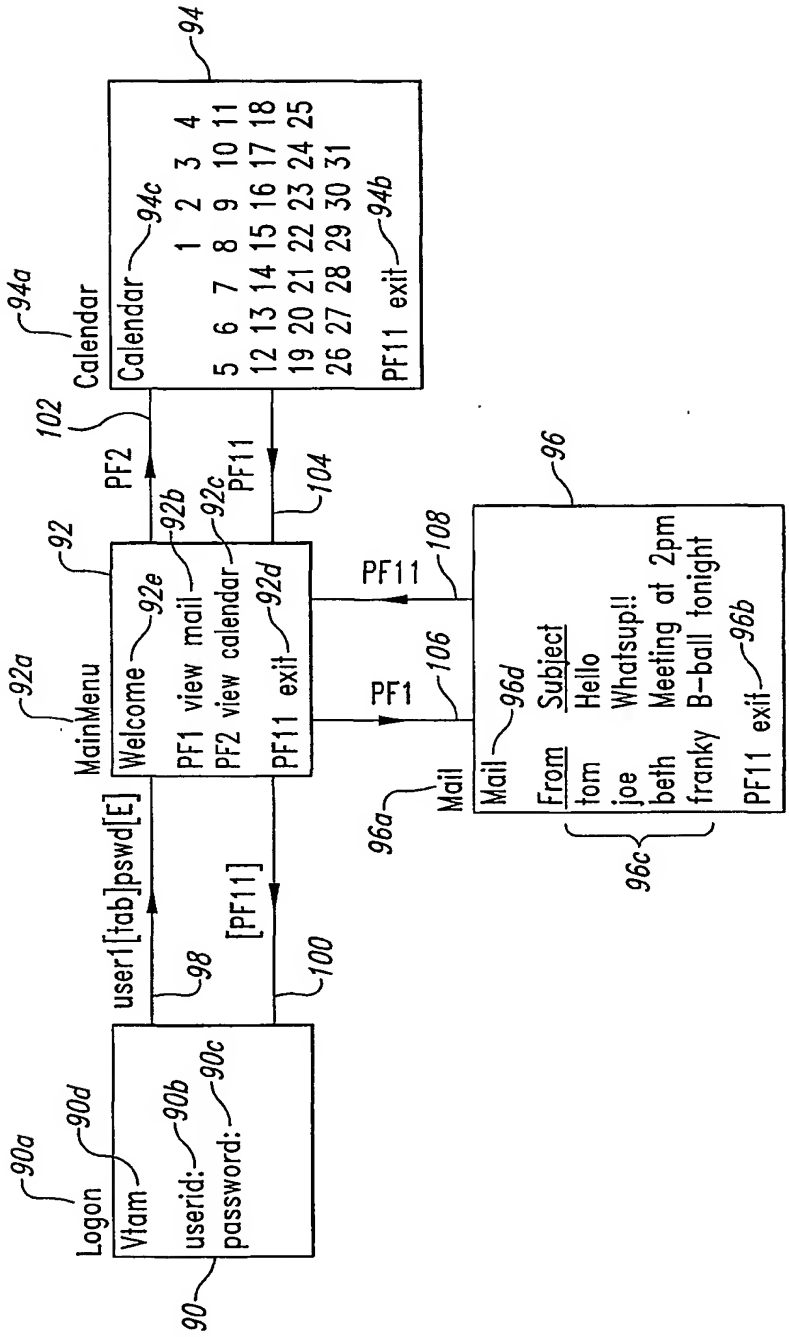

Logon
Vtam
userid:
password:
90a
90b
90c
90d
90
user1[tab]pswd[E]
98
[PF11]
100
MainMenu
Welcome
PF1 view mail
PF2 view calendar
PF11 exit
92a
92b
92c
92d
92e
92
PF2
102
PF11
104
Calendar
Calendar
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
PF11 exit
94a
94b
94c
94
PF11
108
PF1
106
Mail
Mail
From Subject
tom Hello
joe Whatsup!!
beth Meeting at 2pm
franky B-ball tonight
PF11 exit
96a
96b
96c
96d
96

Fig. 3

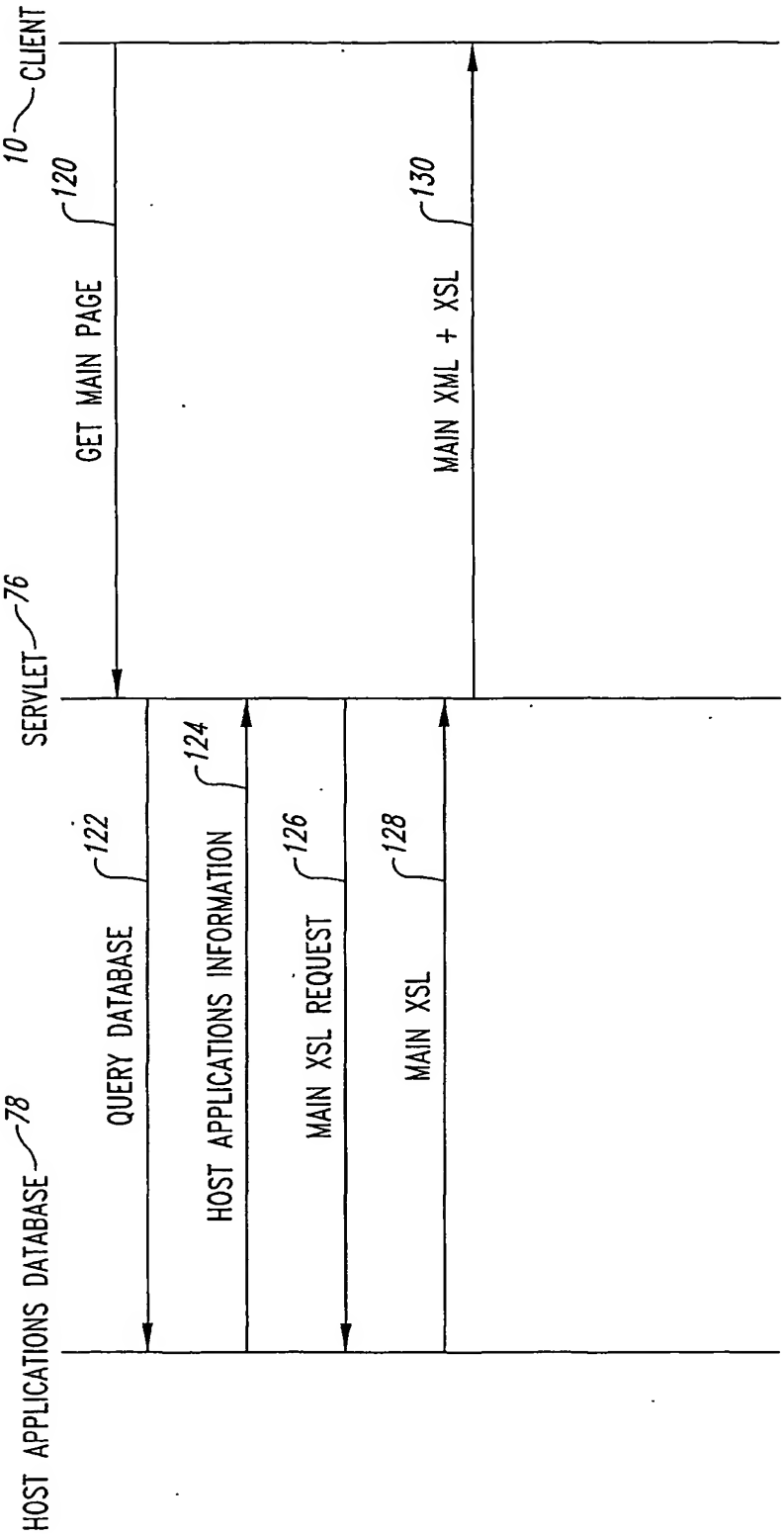

CLIENT
10
120
GET MAIN PAGE
130
MAIN XML + XSL
SERVLET
76
122
QUERY DATABASE
124
HOST APPLICATIONS INFORMATION
126
MAIN XSL REQUEST
128
MAIN XSL
HOST APPLICATIONS DATABASE
78

Fig. 4

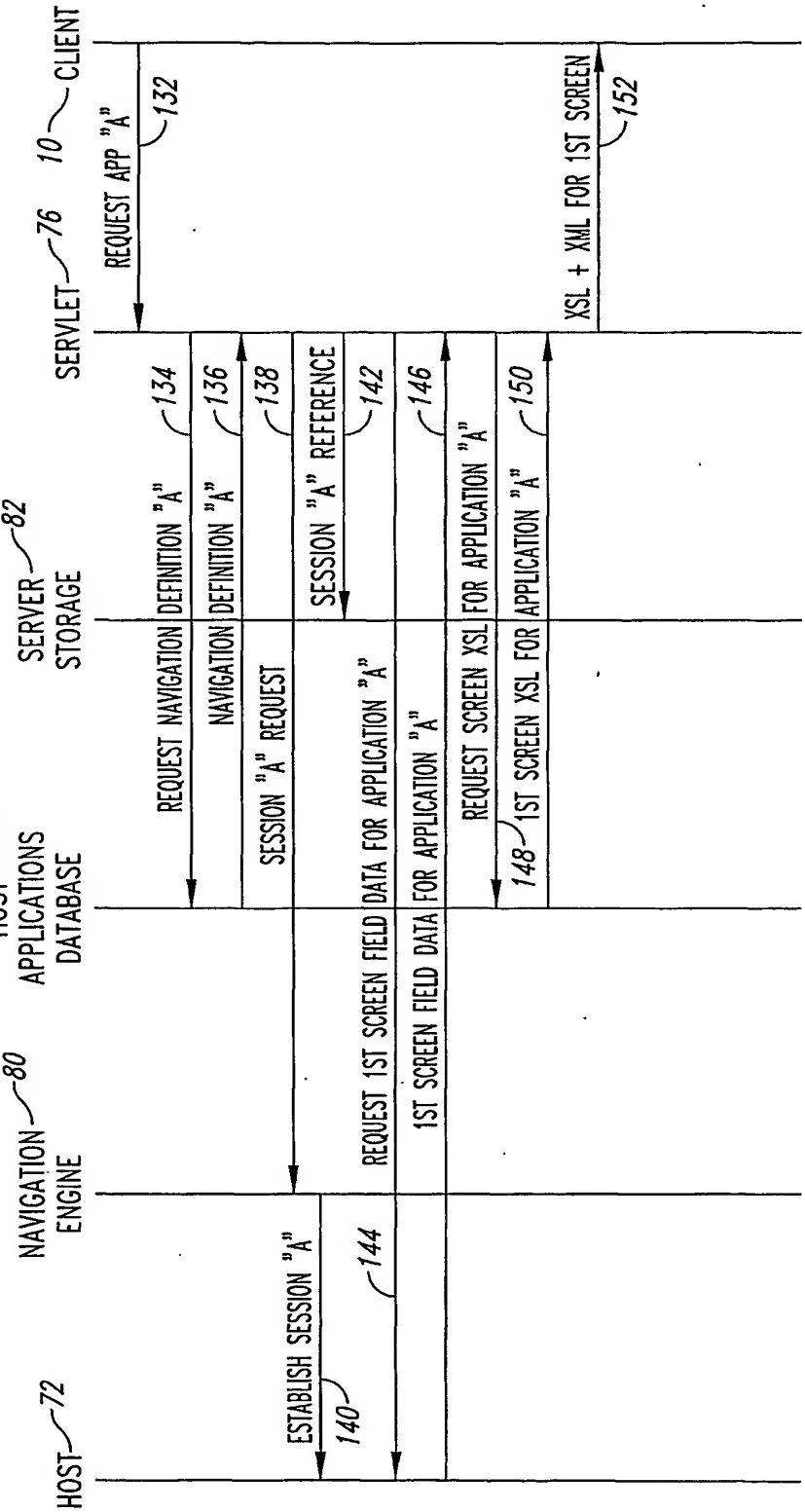

HOST 72
NAVIGATION ENGINE 80
HOST APPLICATIONS DATABASE 78
SERVER STORAGE 82
SERVLET 76
CLIENT 10
REQUEST APP "A" 132
XSL + XML FOR 1ST SCREEN 152
REQUEST NAVIGATION DEFINITION "A" 134
NAVIGATION DEFINITION "A" 136
SESSION "A" REQUEST 138
SESSION "A" REFERENCE 142
DATA FOR APPLICATION "A" 146
REQUEST SCREEN XSL FOR APPLICATION "A" 150
148 1ST SCREEN XSL FOR APPLICATION "A"
ESTABLISH SESSION "A" 140
144
REQUEST 1ST SCREEN FIELD
1ST SCREEN FIELD DATA FOR APPLICATION "A"

Fig. 5

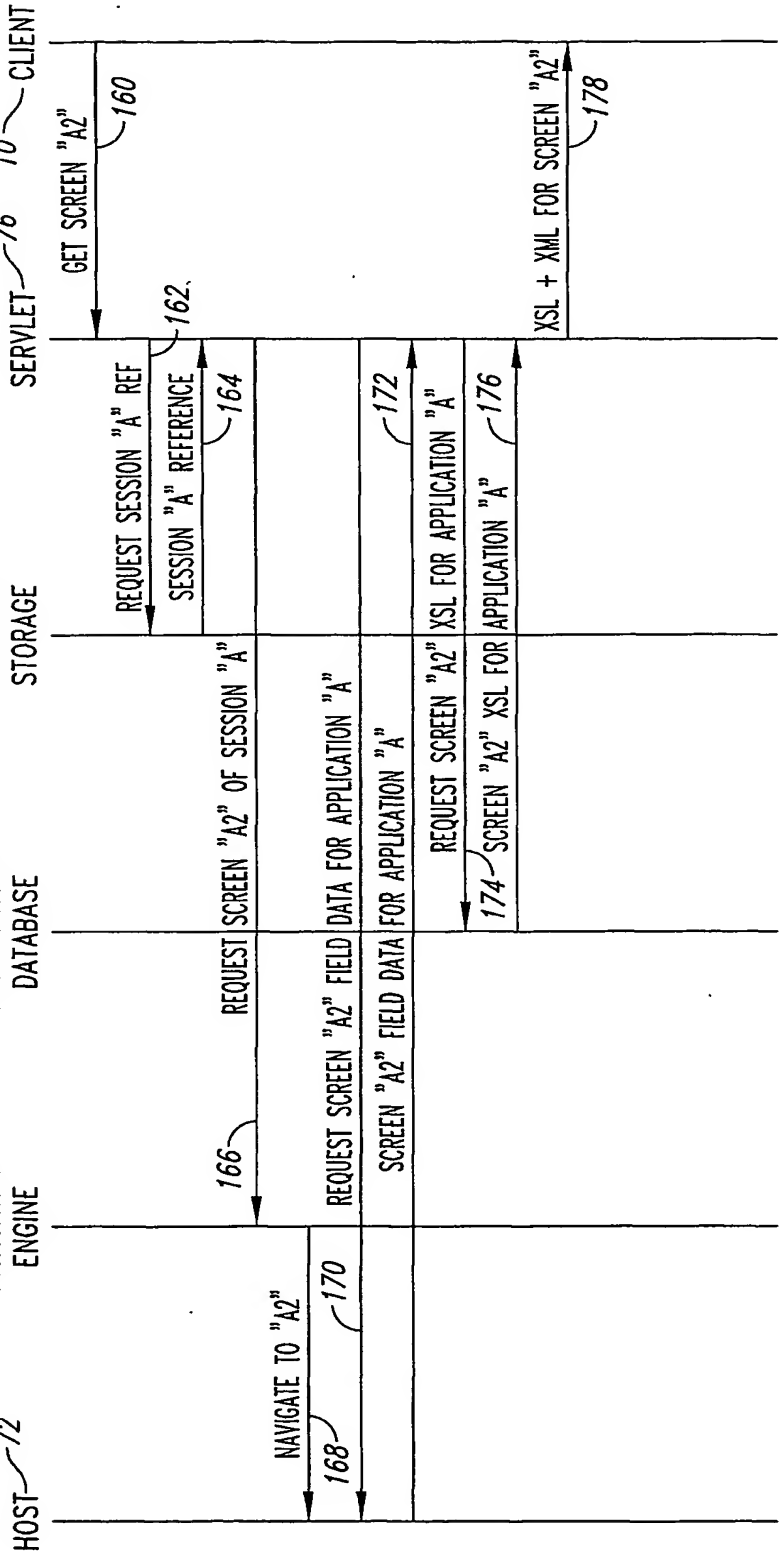

CLIENT 10
SERVLET 76
SERVER STORAGE 82
HOST APPLICATIONS DATABASE 78
NAVIGATION ENGINE 80
HOST 72
GET SCREEN "A2" 160
REQUEST SESSION "A" REF 162
SESSION "A" REFERENCE 164
REQUEST SCREEN "A2" OF SESSION "A" 166
NAVIGATE TO "A2" 168
REQUEST SCREEN "A2" FIELD DATA FOR APPLICATION "A" 170
SCREEN "A2" FIELD DATA FOR APPLICATION "A" 172
REQUEST SCREEN "A2" XSL FOR APPLICATION "A" 174
SCREEN "A2" XSL FOR APPLICATION "A" 176
XSL + XML FOR SCREEN "A2" 178

Fig. 6

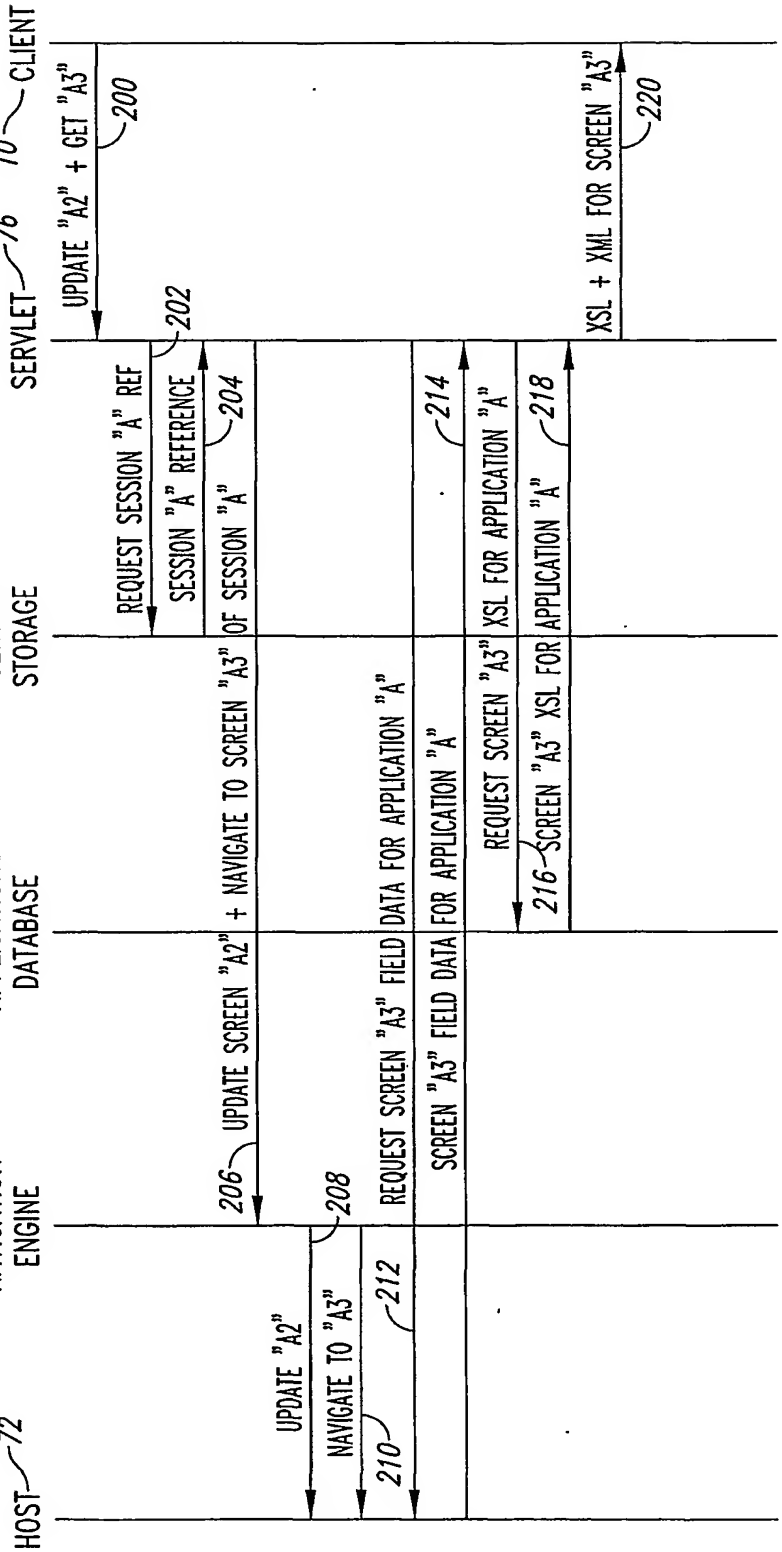

HOST 72
NAVIGATION ENGINE 80
HOST APPLICATIONS DATABASE 78
SERVER STORAGE 82
SERVLET 76
CLIENT 10
UPDATE "A2" + GET "A3" 200
REQUEST SESSION "A" REF 202
SESSION "A" REFERENCE 204
OF SESSION "A"
206 UPDATE SCREEN "A2" + NAVIGATE TO SCREEN "A3"
208 REQUEST SCREEN "A3" FIELD DATA FOR APPLICATION "A"
214
SCREEN "A3" FIELD DATA FOR APPLICATION "A"
REQUEST SCREEN "A3" XSL FOR APPLICATION "A"
216 SCREEN "A3" XSL FOR APPLICATION "A" 218
XSL + XML FOR SCREEN "A3" 220
UPDATE "A2"
NAVIGATE TO "A3"
210
212

Fig. 7

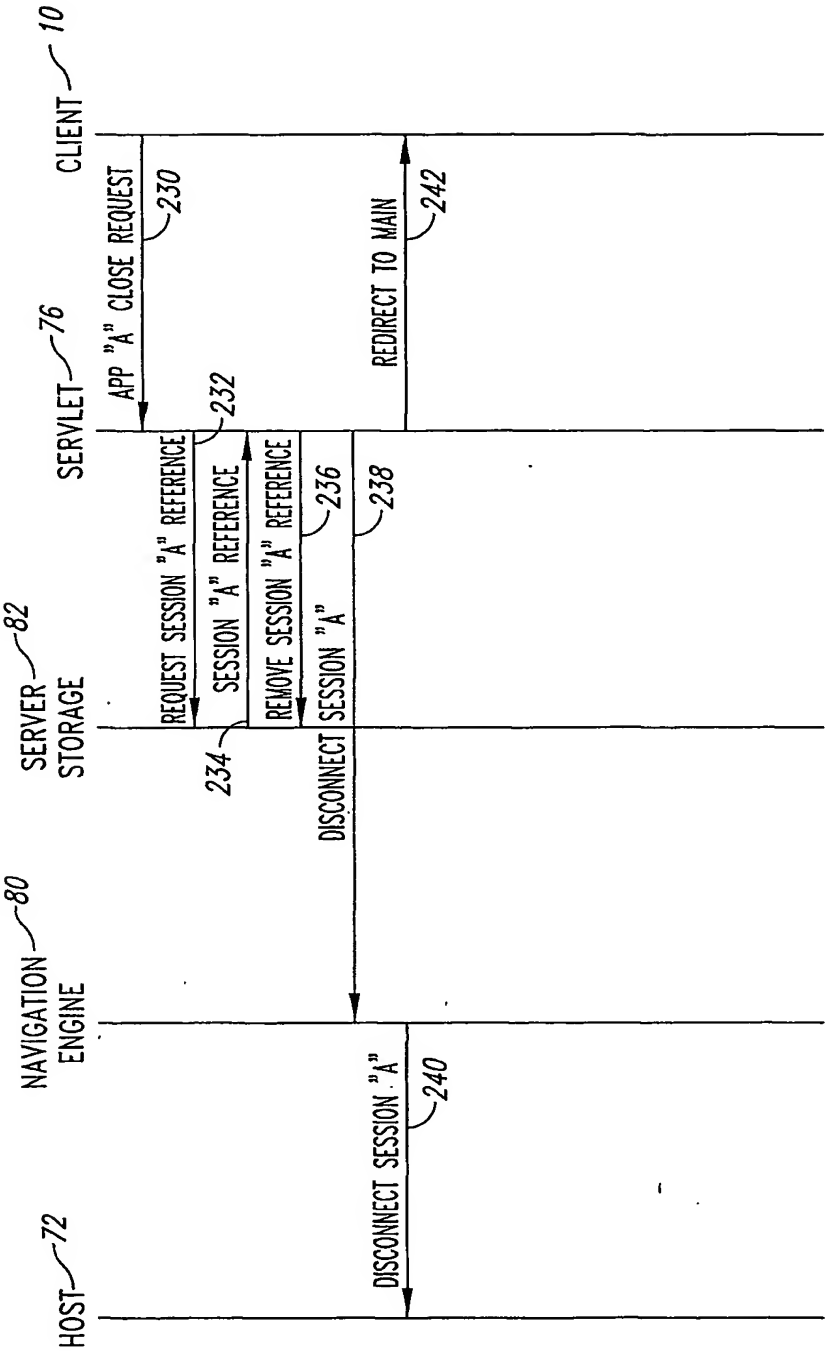

CLIENT
10
SERVLET
76
SERVER STORAGE
82
NAVIGATION ENGINE
80
HOST
72
APP "A" CLOSE REQUEST
230
REQUEST SESSION "A" REFERENCE
232
SESSION "A" REFERENCE
234
REMOVE SESSION "A" REFERENCE
236
DISCONNECT SESSION "A"
238
DISCONNECT SESSION "A"
240
REDIRECT TO MAIN
242

Fig. 8